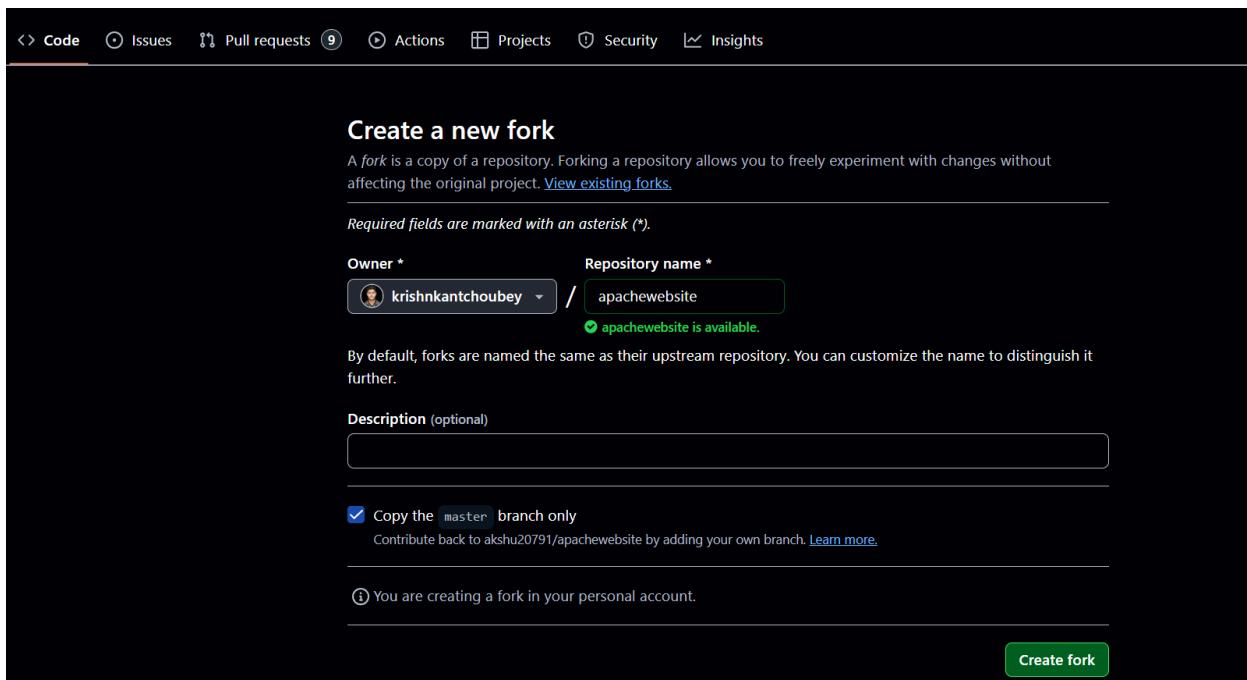


Deploying Apache Project using Ansible via Jenkins

Aim : The project aims to Deploy Apache project on Node EC2's using Ansible which will be triggered via Jenkins pipeline

Git Hub repo to fork :

<https://github.com/akshu20791/apachewebsite>



Git Hub repo for reference :

[krishnkantchoubey/apachewebsite](https://github.com/krishnkantchoubey/apachewebsite)

The screenshot shows a GitHub repository page for 'apachewebsite'. The repository is public and was forked from 'akshu20791/apachewebsite'. The 'Code' tab is selected. The repository has 1 branch and 0 tags. A message indicates that the branch is up-to-date with the original repository's master branch. The commit history shows 30 commits from 'akshu20791' over the past 4 months, with updates to index.html, Dockerfile, buildspec.yml, and test files. The right sidebar provides information about the repository, including activity (0 stars, 0 forks), releases (none published), and packages (none published).

Author	Commit Message	Date
akshu20791	Update index.html	50d7bbf · 4 months ago
	apache web	3 years ago
	apache web	3 years ago
	apache web	3 years ago
	apache web	3 years ago
	Update Dockerfile	10 months ago
	Update buildspec.yml	last year
	Update index.html	4 months ago
	Update test	7 months ago

Base plan :

- 1.Create a Master EC2 with – Jenkins, Ansible, docker installed
2. Create Cluster and nodes
- 3.Connect the Node to master
- 4.Make a Jenkins pipeline and deploy Apache project on all nodes by running Ansible playbook

Steps to Follow:

- 1. Create a Master EC2 Instance**
- 2. Create an IAM User**
- 3. Install Required Packages and Tools**

4. Create the EKS Cluster

5. Configure and Establish Jenkins

6. Install Necessary Plugins in Jenkins

7. Set Up Dynamic Inventory for Ansible

8. Connect Ansible Master with Nodes

9. Execute the Final Deployment Using Jenkins Pipeline

10. Verify and Confirm Successful Deployment

Step 1. Create a Master EC2

a) Give name to EC2 and select AMI as Ubuntu 24.04LTS

The screenshot shows the 'Launch an instance' wizard. In the 'Name and tags' section, the 'Name' field contains 'Krishna-master'. In the 'Application and OS Images (Amazon Machine Image)' section, the search bar is empty. Below the search bar, there's a note about using the search field or 'Browse more AMIs'. At the bottom, there are 'Recents' and 'Quick Start' buttons.

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

Krishna-master

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Search our full catalog including 1000s of application and OS images

Recents **Quick Start**

Amazon Linux macOS Ubuntu Windows Red Hat SUSE Linux Debi:
      

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type Free tier eligible ▾
 ami-0360c520857e3138f (64-bit (x86)) / ami-026fcdd88446aa0bf (64-bit (Arm))
 Virtualization: hvm ENA enabled: true Root device type: ebs

Description
 Ubuntu Server 24.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
 Canonical, Ubuntu, 24.04, amd64 noble image

b) Select instance type as t2.large (since this is master EC2 it will require large size)

▼ **Instance type** [Info](#) | [Get advice](#)

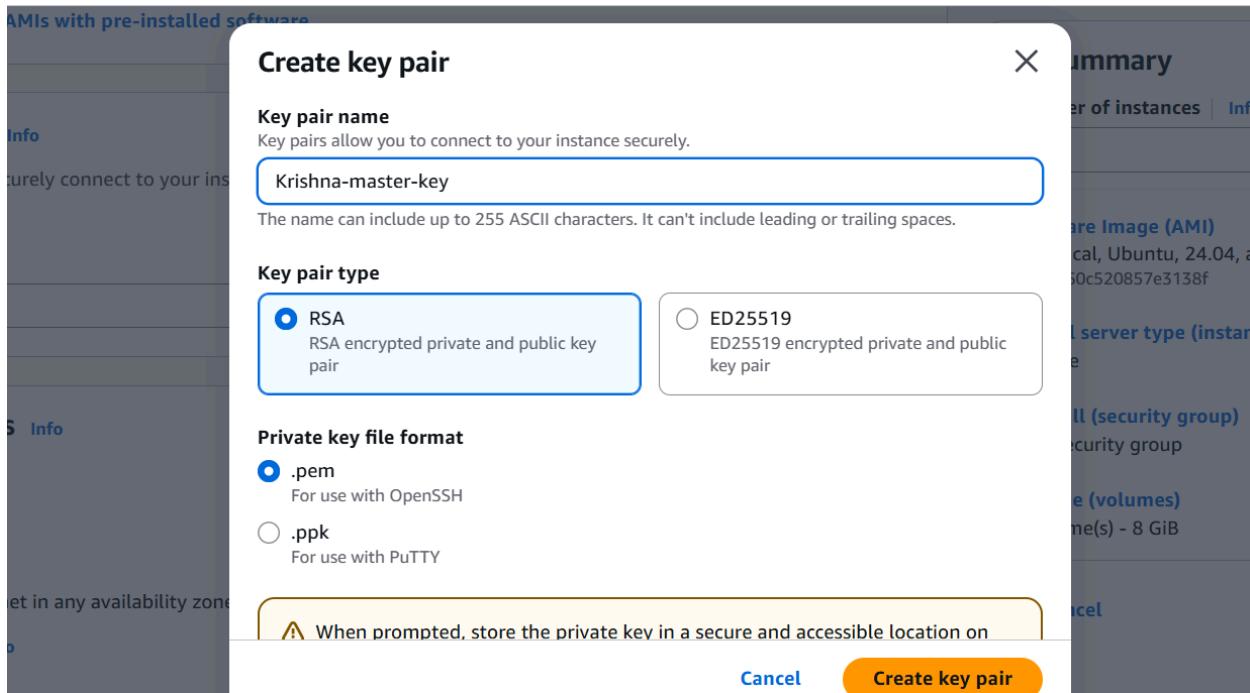
Instance type

t2.large
 Family: t2 2 vCPU 8 GiB Memory Current generation: true
 On-Demand Windows base pricing: 0.1208 USD per Hour
 On-Demand RHEL base pricing: 0.1216 USD per Hour
 On-Demand SUSE base pricing: 0.1928 USD per Hour
 On-Demand Ubuntu Pro base pricing: 0.0963 USD per Hour
 On-Demand Linux base pricing: 0.0928 USD per Hour

All generations [Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

c) Create a Key pair (Save it we will use this key pair throughout the project)
 Give it name -> key pair type = RSA -> Key file format = .pem
 -> Create key pair



d) Create Security Group

1. Click on **Create Security Group**.

2. Enter the following details:

- **Name:** Provide a suitable name for the security group.
- **Description:** Enter a brief description.
- **VPC:** Select the **default VPC**.

3. Add the following **Inbound rules**:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	Allow SSH access
HTTP	TCP	80	0.0.0.0/0	Allow web traffic
HTTPS	TCP	443	0.0.0.0/0	Allow secure web traffic
Custom TCP Rule	TCP	6443	0.0.0.0/0	For Kubernetes API access
Custom TCP Rule	TCP	8080	0.0.0.0/0	For Jenkins access

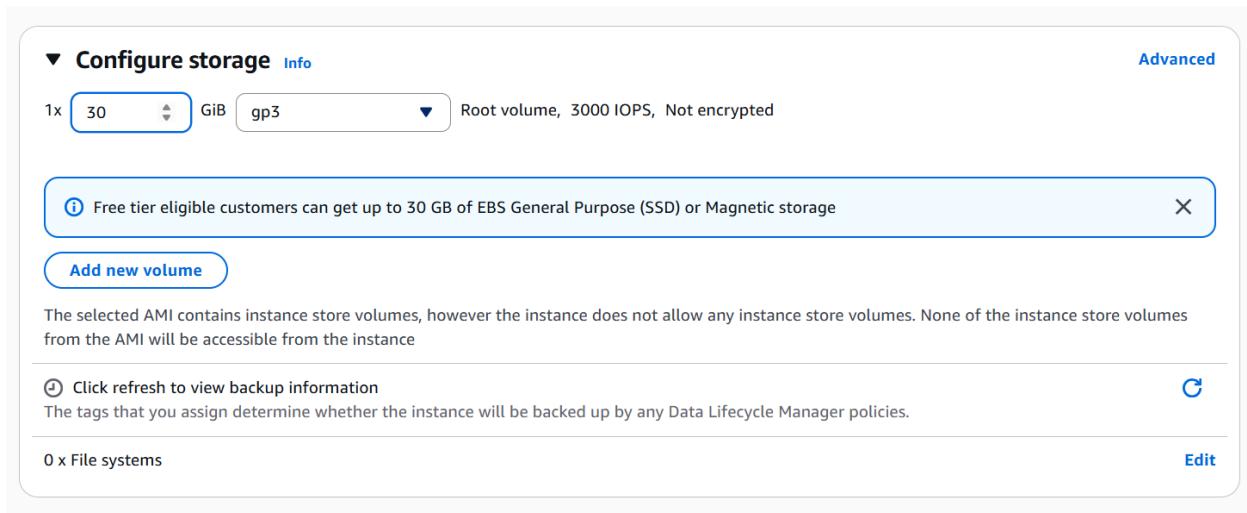
4. Click **Create Security Group** to finalize

The screenshot shows the 'Edit inbound rules' section of the AWS EC2 Security Groups interface. The table lists five rules:

- SSH (TCP port 22) - Source: 0.0.0.0/0
- HTTP (TCP port 80) - Source: 0.0.0.0/0
- HTTPS (TCP port 443) - Source: 0.0.0.0/0
- Custom TCP (TCP port 6443) - Source: 0.0.0.0/0
- Custom TCP (TCP port 8080) - Source: 0.0.0.0/0

Each rule has a 'Delete' button next to it. At the bottom left, there is a blue 'Add rule' button.

e) Configure Storage as 1x30 GiB gp3 then click on Launch Instance



Step 2 . Create IAM User

User name	Path	Groups	Last activity	MFA	Password age	Consoles
Aditi-eks-cluster	/	0	2 days ago	-	7 days	-
akshat-user	/	0	2 days ago	-	-	April
aswini-eks-cluster	/	0	2 days ago	-	7 days	-
crossplane1	/	2	2 days ago	-	-	April
crossplane2	/	1	2 days ago	-	156 days	April

b) Create IAM User

1. Enter the **Name** for the IAM user.
2. Check **Provide user access**.
3. Select **I want to create an IAM user**.
4. Click **Custom password** and enter the desired password.
5. Uncheck **User must create a new password at next sign-in**.
6. Click **Next**.
7. Skip all additional settings and click **Next** again.
8. Click **Create user** to complete the process.

EC2

IAM > Users > Create user

Step 1
 Specify user details
 Step 2
 Set permissions
 Step 3
 Review and create
 Step 4
 Retrieve password

Specify user details

User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = . @ _ - (hyphen)

Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice [to manage their access in IAM Identity Center](#).

Are you providing console access to a person?

User type

Specify a user in Identity Center - Recommended
We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

I want to create an IAM user
We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keypairs, or a backup credential for emergency account access.

Console password

EC2

IAM > Users > Create user

Console password

Autogenerated password
You can view the password after you create the user.

Custom password
Enter a custom password for the user.

Show password

Users must create a new password at next sign-in - Recommended
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keypairs, you can generate them after you create this IAM user. [Learn more](#)

Cancel **Next**

c) Move back to user list and select your user -> click Add Permissions -> Click on Attach Policies Directly

Permissions options

- Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1499)

Choose one or more policies to attach to your new user.

Policy name	Type	Attached entities
AccessAnalyzerServiceRolePolicy	AWS managed	0

User details

User name Krishna-Project	Console password type Autogenerated	Require password reset Yes
------------------------------	--	-------------------------------

Permissions summary

Name	Type	Used as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEKSClusterPolicy	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy
AWSCloudFormationFullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy
IAMUserChangePassword	AWS managed	Permissions policy

Click on Add Permission -> Create inline policy

Summary

ARN arn:aws:iam::430118834478:user/Krishna-Project	Console access ⚠ Enabled without MFA	Access key 1 Create access key
Created September 11, 2025, 09:16 (UTC+05:30)	Last console sign-in ⓘ Never	

Permissions | Groups | Tags | Security credentials | Last Accessed

Permissions policies (7)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type

Search | All types ▾

Policy name ▾ | Type ▾ | Attached via ▾

Add permissions ▾
Add permissions | Create inline policy

IAM > Users > Krishna-Project > Create policy

Step 1 Specify permissions

Step 2 Review and create

Specify permissions Info

Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.

Policy editor

```

1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": "eks:*",
8       "Resource": "*"
9     }
10   ]
11 }
12

```

Visual | **JSON** | Actions ▾

Edit statement

Select a statement

Select an existing statement in the policy
add a new statement.

+ Add new statement

EC2

IAM > Users > Krishna-Project > Create policy

Step 1 Specify permissions

Step 2 Review and create

Review and create Info

Review the permissions, specify details, and tags.

Policy details

Policy name
Enter a meaningful name to identify this policy.

Maximum 128 characters. Use alphanumeric and '+-=_,@-' characters.

Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 450 services)

Show remaining 449 services

Service	▲ Access level	▼ Resource	Request condition
---------	------------------	--------------	-------------------

Now you will have 8 policies attached to user

The screenshot shows the AWS IAM Policies page. The top navigation bar includes EC2, IAM, Users, and Krishna-Project. The left sidebar has sections for Identity and Access Management (IAM), Access management (User groups, Roles, Policies, Identity providers, Account settings, Root access management), and Access reports (Access Analyzer). The main content area displays a table titled "Policy Krishna-Project-policy created." with the following data:

Policy name	Type	Attached via
AmazonEC2FullAccess	AWS managed	Directly
AmazonEKS_CNI_Policy	AWS managed	Directly
AmazonEKSClusterPolicy	AWS managed	Directly
AmazonEKSWorkerNodePolicy	AWS managed	Directly
AWSCloudFormationFullAccess	AWS managed	Directly
IAMFullAccess	AWS managed	Directly
IAMUserChangePassword	AWS managed	Directly
Krishna-Project-policy	Customer inline	Inline

Below the table, a note says "▶ Permissions boundary (not set)".

Step 3. Installations

a) Connect EC2 to ssh

The screenshot shows the AWS EC2 Instances page. The top navigation bar includes EC2 and Instances. The left sidebar has sections for EC2 (Dashboard, EC2 Global View, Events), Instances (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), and Images (AMIs, AMI Catalog). The main content area displays a table titled "Instances (1/6) Info" with the following data:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Z
Krishna-master	i-0de2013047cbd9de3	Running	t2.large	2/2 checks passed	View alarms +	us-east-1b
Tirumala-master	i-00e4a619fa5cd8ce2	Running	t3.medium	Initializing	View alarms +	us-east-1b
Tirumala-node1	i-09aaa591e7b8ab72	Running	t3.medium	Initializing	View alarms +	us-east-1b

Below the table, a detailed view is shown for the instance "i-0de2013047cbd9de3 (Krishna-master)". It includes tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. Under Details, it shows Instance ID (i-0de2013047cbd9de3), Public IPv4 address (54.91.43.234), Private IPv4 addresses (172.31.31.194), and Public DNS.

EC2 > Instances > i-0de2013047cbd9de3 > Connect to instance

Connect Info

Connect to an instance using the browser-based client.

EC2 Instance Connect Session Manager SSH client EC2 serial console

Instance ID i-0de2013047cbd9de3 (Krishna-master)

Connection type Connect using a Public IP Connect using a public IPv4 or IPv6 address Connect using a Private IP Connect using a private IP address and a VPC endpoint

Public IPv4 address 54.91.43.234
 IPv6 address

Username

Note: In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI source has changed the default AMI username.

b) Run basic commands

sudo su

apt update -y

```
ubuntu@ip-172-31-31-194:~$ sudo su
root@ip-172-31-31-194:/home/ubuntu# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease

i-0de2013047cbd9de3 (Krishna-master)
Public IPs: 54.91.43.234 Private IPs: 172.31.31.194
CloudShell Feedback © 2025
```

c) Jenkins vi

Jenkins.sh

#!/bin/bash

Install OpenJDK 17 JRE Headless

sudo apt install openjdk-17-jre-headless -y

```
# Download Jenkins GPG key
```

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \  
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

```
# Add Jenkins repository to package manager sources
```

```
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  
https://pkg.jenkins.io/debian-stable binary/" | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
# Update package manager repositories
```

```
sudo apt-get update
```

```
# Install Jenkins
```

```
sudo apt-get install jenkins -y
```

```
total 4  
-rw-r--r-- 1 root root 568 Sep 11 05:52 Jenkins.sh  
root@ip-172-31-31-194:/home/ubuntu# chmod +x Jenkins.sh  
root@ip-172-31-31-194:/home/ubuntu# ./Jenkins.sh  
Reading package lists... Done
```

```
i-0de2013047cbd9de3 (Krishna-master)
```

```
Public IPs: 54.91.43.234 Private IPs: 172.31.31.194
```

```
press - esc :wq
```

```
give permissions and run the file
```

```
chmod +x jenkin.sh
```

```
./jenkins.sh
```

```
Running kernel seems to be up-to-date.  
No services need to be restarted.  
No containers need to be restarted.  
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell 

Check for java and Jenkins working

java -version

```
root@ip-172-31-31-194:/home/ubuntu# java --version  
openjdk 17.0.16 2025-07-15  
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu124.04.1)  
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)  
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell 

sudo systemctl status jenkins

```
root@ip-172-31-31-194:/home/ubuntu# sudo systemctl status jenkins  
● jenkins.service - Jenkins Continuous Integration Server  
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)  
   Active: active (running) since Thu 2025-09-11 05:54:34 UTC; 6min ago  
     Main PID: 4858 (java)  
        Tasks: 45 (limit: 9498)  
       Memory: 981.3M (peak: 997.8M)  
          CPU: 17.971s  
        CGroup: /system.slice/jenkins.service  
               └─4858 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080  
  
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: e18a2d64750f4521883dael287729e2d  
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword  
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ****  
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ****  
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ****  
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.268+0000 [id=30]      INFO      jenkins.InitReactorRunner$1#onAttained: Comp  
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.284+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins  
Sep 11 05:54:34 ip-172-31-31-194 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.  
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.441+0000 [id=49]      INFO      h.m.DownloadService$Downloadable#load: Obtain  
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.451+0000 [id=49]      INFO      hudson.util.Retriger#start: Performed the act  
lines 1-20/20 (END)
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell 

 **Setup Jenkins**

Open Port 8080 in the Jenkins Server
Access Jenkins by entering the following in your web browser:

http://<ip_address>:8080

- 1. Replace <ip_address> with the public IP address of your Jenkins server.**

Retrieve the Initial Administrator Password

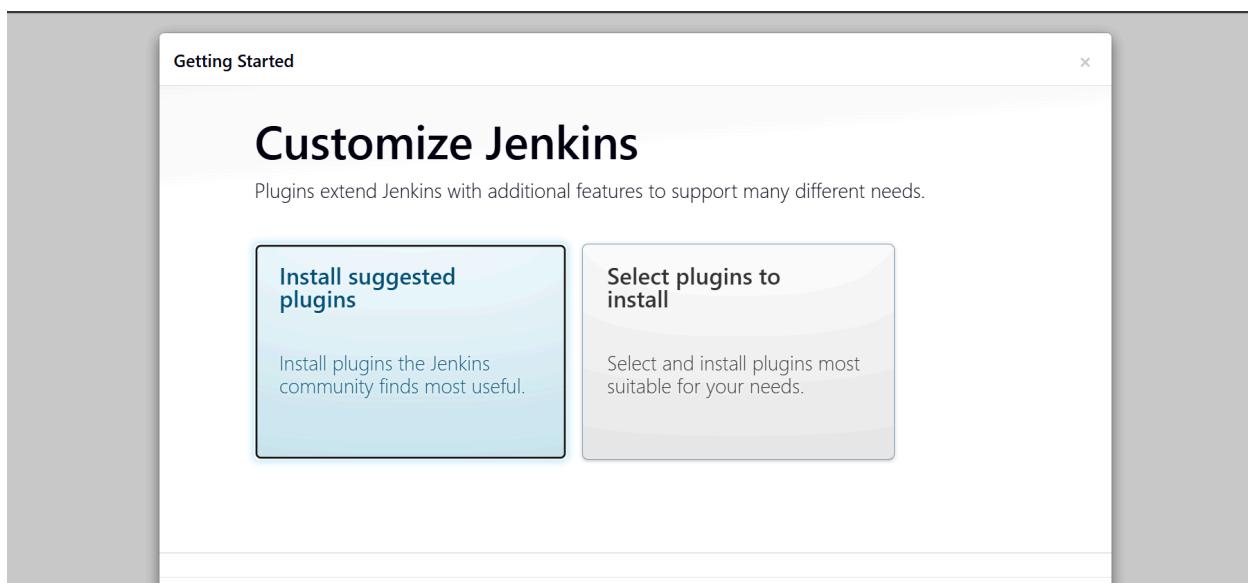
Run the following command to view the password:

cat /var/lib/jenkins/secrets/initialAdminPassword

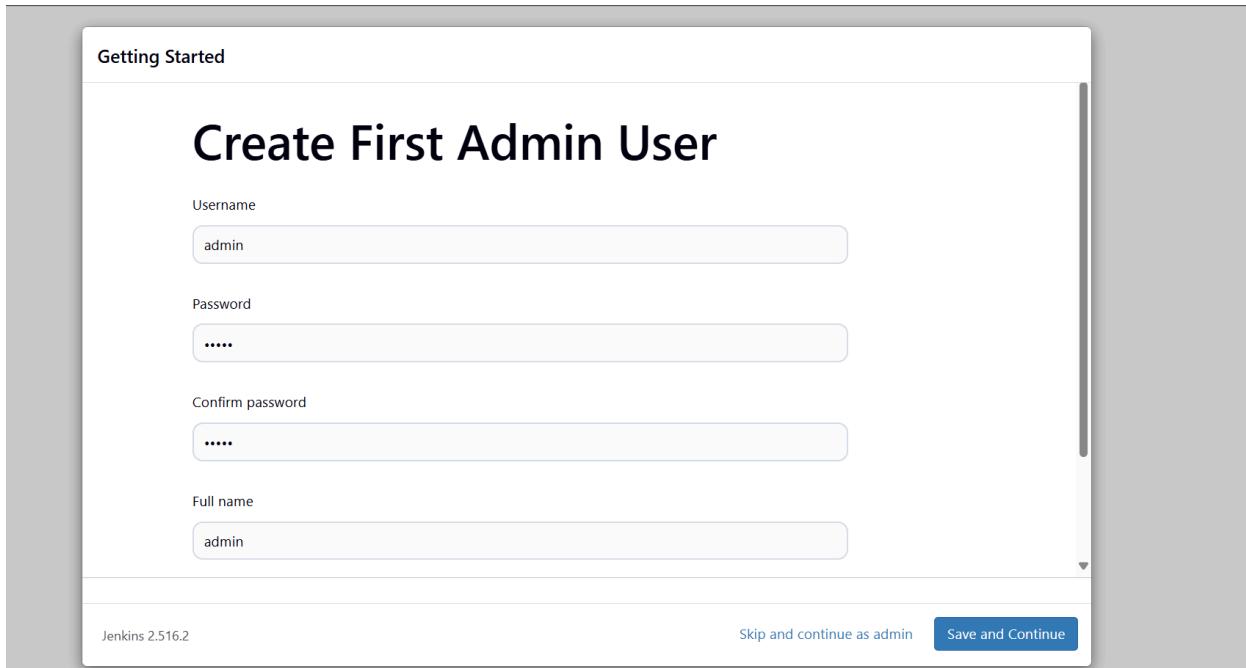
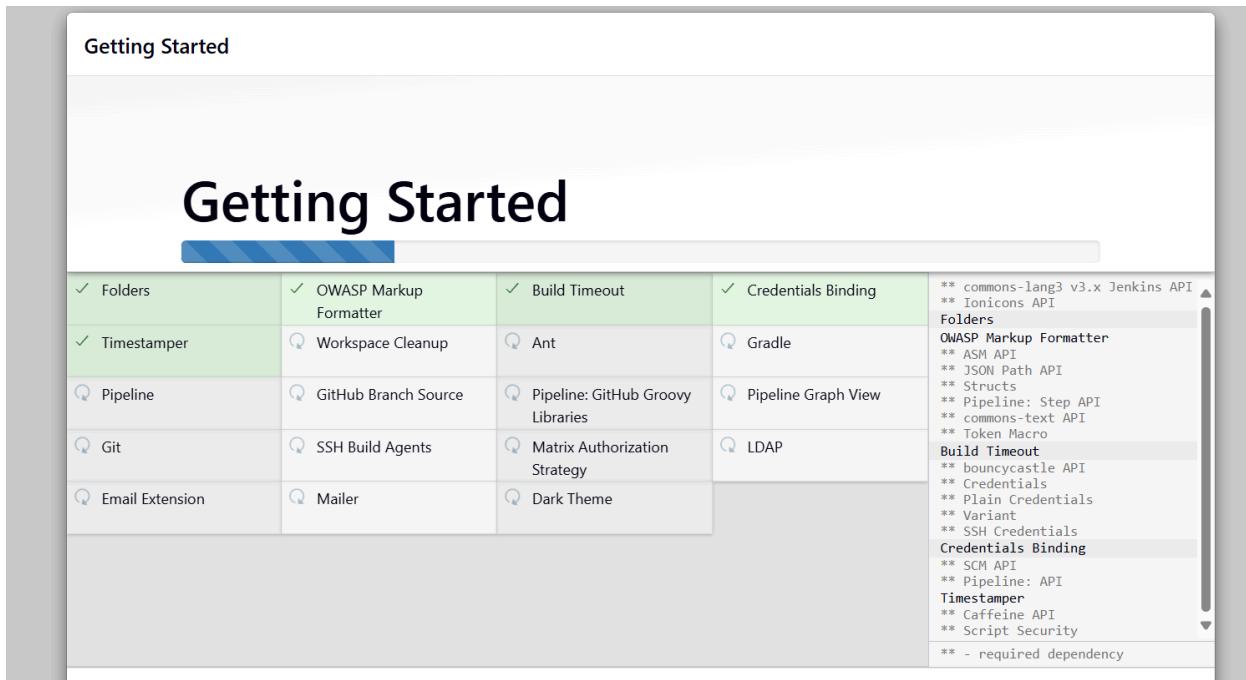
2.

3. Login to Jenkins

Copy the displayed password and paste it into the Administrator password field on the Jenkins setup page.



Click on install all suggested plugins



d) Docker

#!/bin/bash

Update package manager repositories

```
sudo apt-get update

# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo \"${VERSION_CODENAME}\") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories
```

```
sudo apt-get update
```

Install Docker packages

```
sudo apt-get install -y docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

```
# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories
sudo apt-get update

# Install Docker packages
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

```
-- INSERT --

i-0de2013047cbd9de3 (Krishna-master)
Public IPs: 54.91.43.234 Private IPs: 172.31.31.194
```

press – esc :wq

give permission and run the file

chmod +x docker.sh

./docker.sh

```
root@ip-172-31-31-194:/home/ubuntu# cat /var/lib/jenkins/secrets/initialAdminPassword
e18a2d64750f4521883dae1287729e2d
root@ip-172-31-31-194:/home/ubuntu# vi docker.sh
root@ip-172-31-31-194:/home/ubuntu# chmod +x docker.sh
root@ip-172-31-31-194:/home/ubuntu# ./docker.sh
```

```
i-0de2013047cbd9de3 (Krishna-master)
```

```
Public IPs: 54.91.43.234 Private IPs: 172.31.31.194
```

Docker --version

```
root@ip-172-31-31-194:/home/ubuntu# docker --version
Docker version 28.4.0, build d8eb465
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

 CloudShell Feedback

Pull a Default Docker Image

Pull the **hello-world** image by running the following command:

docker pull hello-world

```
DECKER VERSION 28.4.0, Build d8eb465
root@ip-172-31-31-194:/home/ubuntu# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:54e66cc1dd1fcbb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

1.If you see a permission denied error, execute the following command to allow Docker access:

sudo chmod 666 /var/run/docker.sock

2.After changing the permissions, pull the image again:

docker pull hello-world

```
root@ip-172-31-31-194:/home/ubuntu# sudo chmod 666 /var/run/docker.sock
root@ip-172-31-31-194:/home/ubuntu# docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:54e66cc1dd1fcbb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

```
root@ip-172-31-31-194:/home/ubuntu# docker rmi hello-world:latest
Untagged: hello-world:latest
Untagged: hello-world@sha256:54e66cc1dd1fcb1c3c58bd8017914dbed8701e2d8c74d9262e26bd9cc1642d31
Deleted: sha256:1b44b5a3e06a9aae883e7bf25e45c100be0bb81a0e01b32de604f3ac44711634
Deleted: sha256:53d204b3dc5ddbc129df4ce71996b8168711e211274c785de5e0d4eb68ec3851
root@ip-172-31-31-194:/home/ubuntu# docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
root@ip-172-31-31-194:/home/ubuntu# docker login -u krishnakant334

[Info → A Personal Access Token (PAT) can be used instead.  
To create a PAT, visit https://app.docker.com/settings

Password:
WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-172-31-31-194:/home/ubuntu# 
```

i-0de2013047cbd9de3 (Krishna-master)
Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

Remove the Docker Image and Login to Docker Hub

Remove the **hello-world** image by running:

docker rmi hello-world:latest

1. Verify that no images remain by running:

docker images

2. ➤ The output should show no images.

3. Login to Docker Hub

- Open Docker Hub in your browser and log in with your account.
 - This step is necessary; otherwise, the next steps may result in errors.

Login from the terminal by running:

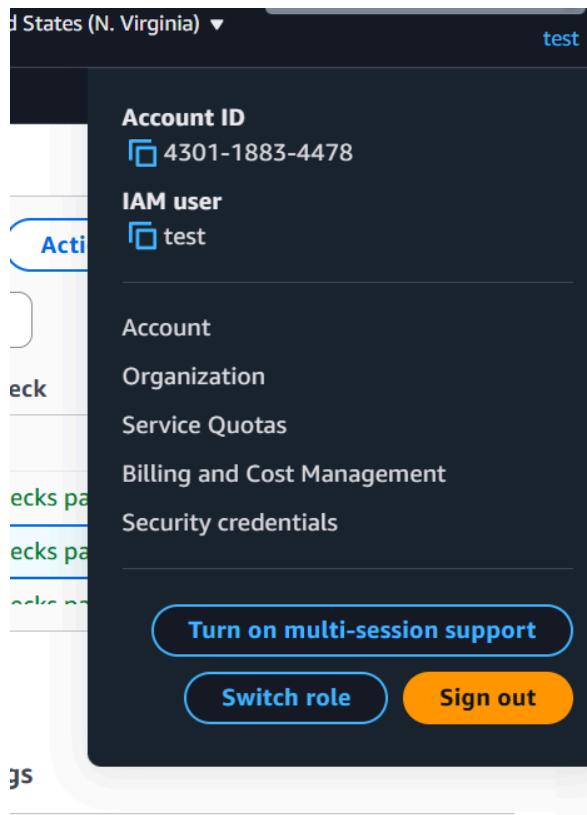
```
docker login -u krishnakant334
```

4. ➤ Press Enter, then type your Docker Hub password and press Enter.

e) AWS cli

First create Access key and Secret Access key

Go to security credentials



Scroll down and click on create access key and create one since I am using sir's account thus I will use the given access key

IAM > Security credentials > Create access key

Step 1
 Access key best practices & alternatives
 Step 2 - *optional*
 Set description tag
 Step 3
 Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

IAM > Security credentials > Create access key

Step 1
 Access key best practices & alternatives
 Step 2 - *optional*
 Set description tag
 Step 3
 Retrieve access keys

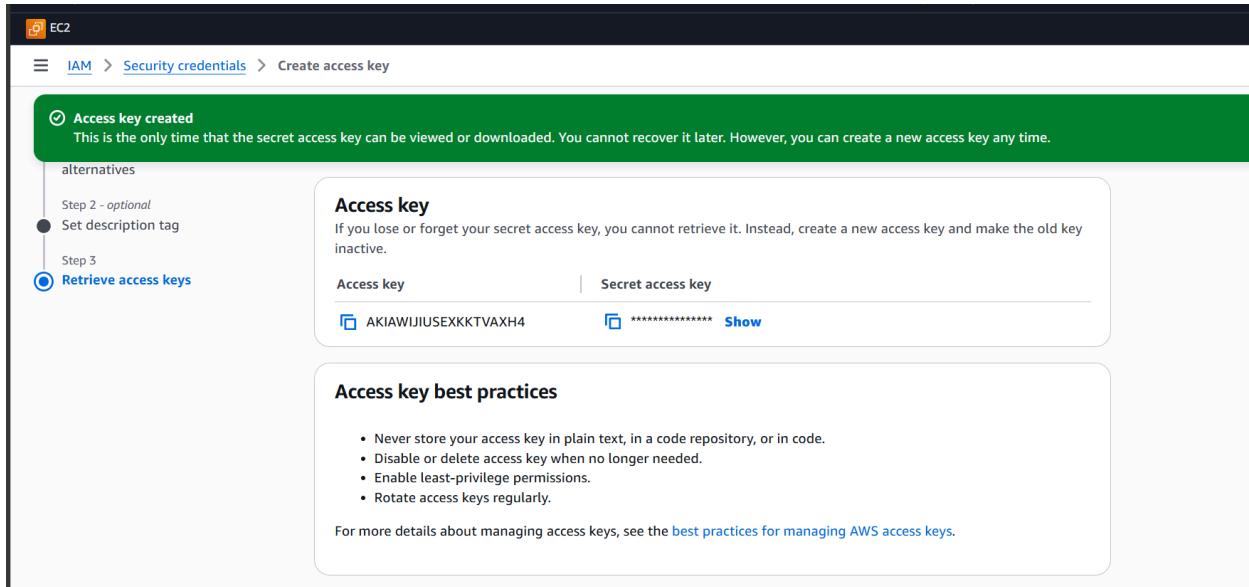
Set description tag - *optional* Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Create access key



Install AWS CLI on the Server

Create the script file by running:

```
vi aws.sh
```

1. Add the following commands inside the file:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"  
sudo apt install unzip -y  
unzip awscliv2.zip  
sudo ./aws/install
```

2. Save and exit the file:

- Press Esc, type :wq, and press Enter.

2. Give execute permissions to the script:

```
chmod +x aws.sh
```

3. Run the script:

```
./aws.sh
```

Install AWS CLI using the downloaded package

Download the AWS CLI installer:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"
```

1.

Install `unzip` if it's not already installed:

```
sudo apt install unzip -y
```

2.

Extract the AWS CLI installer:

```
unzip awscliv2.zip
```

3.

4. Run the installation script:

```
sudo ./aws/install
```

```
Login Succeeded  
root@ip-172-31-31-194:/home/ubuntu# vi aws.sh  
root@ip-172-31-31-194:/home/ubuntu# █
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell  Feedback

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"  
sudo apt install unzip  
unzip awscliv2.zip  
sudo ./aws/install █  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

<https://docs.docker.com/go/credential-store/>

```
Login Succeeded  
root@ip-172-31-31-194:/home/ubuntu# vi aws.sh  
root@ip-172-31-31-194:/home/ubuntu# ^[[200~chmod +x aws.sh ~^C  
root@ip-172-31-31-194:/home/ubuntu# chmod +x aws.sh  
root@ip-172-31-31-194:/home/ubuntu# ./aws.sh  
curl: option -o: requires parameter
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell  Feedback

Aws version check

aws --version

```
root@ip-172-31-31-194:/home/ubuntu# aws --version
Command 'aws' not found, but can be installed with:
snap install aws-cli  # version 1.42.27, or
apt install awscli   # version 2.14.6-1
See 'snap info aws-cli' for additional versions.
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell [Feedback](#)

Configure AWS CLI

Run the following command to start AWS configuration:

```
aws configure
```

Enter the requested details:

- AWS Access Key ID: *Enter your access key*
- AWS Secret Access Key: *Enter your secret key*
- Default region name: <your_working_region>
- Default output format: json

```
initiating. aws/cli/_wheel-0.45.1.dist-info/entry_points.txt
root@ip-172-31-31-194:/home/ubuntu# sudo ./aws/install
You can now run: /usr/local/bin/aws --version
root@ip-172-31-31-194:/home/ubuntu# aws --version
aws-cli/2.29.1 Python/3.13.7 Linux/6.14.0-1011-aws exe/x86_64.ubuntu.24
root@ip-172-31-31-194:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIAWIJIUSEXKKTVAXH4
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell [Feedback](#)

Install kubectl

Create the script file by running:

```
vi kubectl.sh
```

1.

Add the following commands inside the file:

```
curl -o kubectl
https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
```

```
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin
```

```
kubectl version --short --client
```

2.

3. Save and exit the file:

- Press **Esc**, type **:wq**, and press **Enter**.

Give execute permissions to the script:

```
chmod +x kubectl.sh
```

4.

5. Run the script:

```
./kubectl.sh
```

```
Default output format [None].json
root@ip-172-31-31-194:/home/ubuntu# vi kubectl.sh
root@ip-172-31-31-194:/home/ubuntu# chmod +x kubectl.sh
root@ip-172-31-31-194:/home/ubuntu# ./kubectl.sh
  % Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
               Dload  Upload   Total   Spent    Left  Speed
100 57.4M  100 57.4M    0      0  10.5M      0:00:05  0:00:05  ---:-- 11.8M
Client Version: v1.19.6-eks-49a6c0
root@ip-172-31-31-194:/home/ubuntu#
```

i-Ode2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

 CloudShell  Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Install eksctl

Create the script file by running:

```
vi eksctl.sh
```

1.

Add the following commands inside the file:

```
curl --silent --location  
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname  
-s)_amd64.tar.gz" | tar xz -C /tmp
```

```
sudo mv /tmp/eksctl /usr/local/bin
```

```
eksctl version
```

2.

3. Save and exit the file:

- Press **Esc**, type :**wq**, and press **Enter**.

Give execute permissions to the script:

```
chmod +x eksctl.sh
```

4.

5. Run the script:

```
./eksctl.sh
```

```
root@ip-172-31-31-194:/home/ubuntu# vi eksctl.sh  
root@ip-172-31-31-194:/home/ubuntu# chmod +x eksctl.sh  
root@ip-172-31-31-194:/home/ubuntu# ./eksctl.sh  
0.214.0  
root@ip-172-31-31-194:/home/ubuntu# █
```

i-Ode2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

 CloudShell  Feedback

Install Ansible

Create the script file by running:

```
vi ansible.sh
```

1.

Add the following commands inside the file:

```
sudo add-apt-repository --yes --update ppa:ansible/ansible
```

```
sudo apt install ansible -y
```

2.

3. Save and exit the file:

- Press **Esc**, type :**wq**, and press **Enter**.

Give execute permissions to the script:

```
chmod +x ansible.sh
```

4.

Run the script:

```
./ansible.sh
```

5.

6. Check the installed version of Ansible:

```
ansible --version
```

```
root@ip-172-31-31-194:/home/ubuntu# ansible --version
ansible [core 2.18.9]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] (/usr/bin/python3)
  jinja version = 3.1.2
  libyaml = True
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

```
#!/bin/bash

# --- Set your subnets ---
SUBNETS=",<your_subnet_b>"

# --- Create EKS cluster using existing VPC ---
eksctl create cluster --name krishnamaster \
--region up-east-1 \
--version 1.30 \
--without-nodegroup \
--vpc-private-subnets $SUBNETS

~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
-- INSERT --
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

```
#!/bin/bash

# --- Set your subnets ---
SUBNETS="subnet-09527e09d2b636418,subnet-01dfc86302e426722"

# --- Create EKS cluster using existing VPC ---
eksctl create cluster --name krishnamaster \
--region us-east-1 \
--version 1.30 \
--without-nodegroup \
--vpc-private-subnets $SUBNETS
```

Instructions

Save this script to `cluster.sh`:

```
vi cluster.sh
```

1. Paste the above content.
2. Save and exit (`Esc`, type `:wq`, press `Enter`).

Make the script executable:

```
chmod +x cluster.sh
```

- 3.
4. Run the script:

```
./cluster.sh
```

```
*** System restart required ***
Last login: Thu Sep 11 06:49:09 2025 from 18.206.107.27
ubuntu@ip-172-31-31-194:~$ sudo su
root@ip-172-31-31-194:/home/ubuntu# vi cluster.sh
root@ip-172-31-31-194:/home/ubuntu# vi cluster.sh
root@ip-172-31-31-194:/home/ubuntu# vi cluster.sh
^C
root@ip-172-31-31-194:/home/ubuntu# chmod +x cluster.sh
root@ip-172-31-31-194:/home/ubuntu# ./cluster.sh
2025-09-11 09:20:19 [i] eksctl version 0.214.0
2025-09-11 09:20:19 [i] using region us-east-1
2025-09-11 09:20:20 [✓] using existing VPC (vpc-04bd9ec282061593f) and subnets (pi
/24 0 } us-east-1b:{subnet-0ldfc86302e426722 us-east-1b 10.0.2.0/24 0 }] public:map
```

i-Ode2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

 CloudShell [Feedback](#)

```

2025-09-11 09:29:22 [i] creating addon: coredns
2025-09-11 09:29:23 [i] successfully created addon: coredns
2025-09-11 09:29:23 [i] creating addon: metrics-server
2025-09-11 09:29:24 [i] successfully created addon: metrics-server
2025-09-11 09:29:24 [!] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdentityAssociations`, and run `eksctl update addon`
2025-09-11 09:29:24 [i] creating addon: vpc-cni
2025-09-11 09:29:25 [i] successfully created addon: vpc-cni
2025-09-11 09:31:25 [i] waiting for the control plane to become ready
2025-09-11 09:31:26 [V] saved kubeconfig as "/root/.kube/config"
2025-09-11 09:31:26 [i] no tasks
2025-09-11 09:31:26 [V] all EKS cluster resources for "krishnamaster" have been created
2025-09-11 09:31:26 [i] kubectl command should work with "/root/.kube/config", try "kubectl get nodes"
2025-09-11 09:31:26 [V] EKS cluster "krishnamaster" in "us-east-1" region is ready
root@ip-172-31-31-194:/home/ubuntu# 

```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

[CloudShell](#) [Feedback](#) © 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#)

b) Associate IAM OIDC Provider

1. Open the script file:

```
vi ekassociation.sh
```

2. Add the following content inside the file:

```
eksctl utils associate-iam-oidc-provider \
--region us-east-1 \
--cluster krishnamaster \
--approve
```

3. Save and exit the file.

4. Give execute permission and run the script:

```
chmod +x ekassociation.sh
```

```
./ekassociation.sh
```

```
eksctl utils associate-iam-oidc-provider \
--region up-east-1 \
--cluster krishnamaster \
--approve
```

-- INSERT --

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

[CloudShell](#) [Feedback](#)

© 2025, A

```
root@ip-172-31-31-194:/home/ubuntu# vi ekassociation.sh
root@ip-172-31-31-194:/home/ubuntu# chmod +x ekassociation.sh
root@ip-172-31-31-194:/home/ubuntu# ./ekassociation.sh
2025-09-11 09:36:41 [i] will create IAM Open ID Connect provider for cluster "krishnamaster" in "us-east-1"
2025-09-11 09:36:41 [v] created IAM Open ID Connect provider for cluster "krishnamaster" in "us-east-1"
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

Public IPs: 54.91.43.234 Private IPs: 172.31.31.194

[CloudShell](#) [Feedback](#)

© 2025, Amazon Web Services, Inc.

```
#!/bin/bash
```

```
eksctl create nodegroup \
--cluster krishnamaster \
--region us-east-1 \
--name node2 \
--node-type t3.medium \
--nodes 2 \
```

```
--nodes-min 1 \
--nodes-max 2 \
--node-volume-size 20 \
--ssh-access \
--ssh-public-key <your_key_name> \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access \
--node-private-networking \
--subnet-ids subnet-09527e09d2b636418,subnet-01dfc86302e426722 \
--tags "Environment=dev"
```

instructions

Save this script as `createnode.sh`:

```
vi createnode.sh
```

1. Paste the content above, then save and exit (`Esc`, `:wq`, `Enter`).

Make it executable:

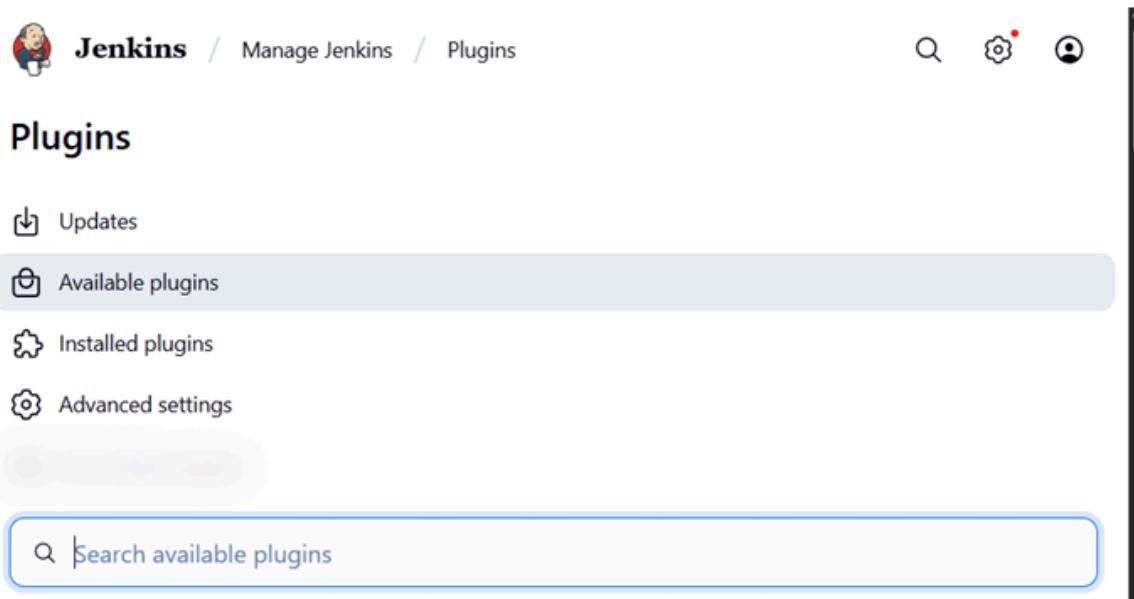
```
chmod +x createnode.sh
```

- 2.
3. Run the script:

```
./createnode.sh
```

Step 5. Establishing Jenkins Plugins

a) Open Jenkins -> Manage Jenkins -> Plugins -> Available Plugins



Install all 13 Plugins listed below and after that restart Jenkins

1. Pipeline: Stage View
2. Docker
3. Docker Commons
4. Docker Pipeline
5. Docker API
6. docker-build-step
7. Kubernetes
8. Kubernetes Client API
9. Kubernetes Credentials
10. Kubernetes CLI
11. Config File Provider
12. Eclipse Temurin Installer (for Java)
13. Ansible



 Jenkins is restarting

Your browser will reload automatically when Jenkins is ready.

Safe Restart
Builds on agents can usually continue.



Plugins

Download progress

Updates

→ [Go back to the top page](#)

(you can start using the installed plugins right away)

Available plugins

→

Restart Jenkins when installation is complete and no jobs are running

Installed plugins

Advanced settings

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

commons-lang3 v3.x Jenkins API	 Success
Ionicons API	 Success
Folders	 Success
OWASP Markup Formatter	 Success
ASM API	 Success
JSON Path API	 Success
Structs	 Success
Pipeline: Step API	 Success
commons-text API	 Success
Token Macro	 Success
Build Timeout	 Success
bouncycastle API	 Success
Credentials	 Success
Plain Credentials	 Success
Variant	 Success
SSH Credentials	 Success
Credentials Binding	 Success
SCM API	 Success
Pipeline: API	 Success
Timestamper	 Success
Caffeine API	 Success

after installation check on Restart option

- [Go back to the top page](#)
(you can start using the installed plugins right away)
- Restart Jenkins when installation is complete and no jobs are running

b) Go to Manage Jenkins -> Credentials (Under security) ->click global -> Add credentials

The screenshot shows the Jenkins 'Credentials' page. At the top, there is a navigation bar with icons for Jenkins, Manage Jenkins, and Credentials, along with search and settings buttons. Below the navigation is a header 'Credentials' with tabs for T, P, Store ↓, Domain, ID, and Name. A sub-header 'Stores scoped to Jenkins' is displayed. Under this, there is a table with columns for P, Store ↓, Domains, Icon, and Name. One row is visible, labeled 'System (global)'. At the bottom left, there are icons for S, M, and L. The URL in the browser is 'http://jenkins:8080/manage/jenkins/store/global'.

The screenshot shows the Jenkins 'Global credentials (unrestricted)' page. At the top, there is a navigation bar with icons for Jenkins, Manage Jenkins, and Global credentials (unrestricted), along with search and settings buttons. Below the navigation is a header 'Global credentials (unrestricted)'. A blue button at the top right says '+ Add Credentials'. The URL in the browser is 'http://jenkins:8080/manage/jenkins/store/global/unrestricted'.

Make below changes

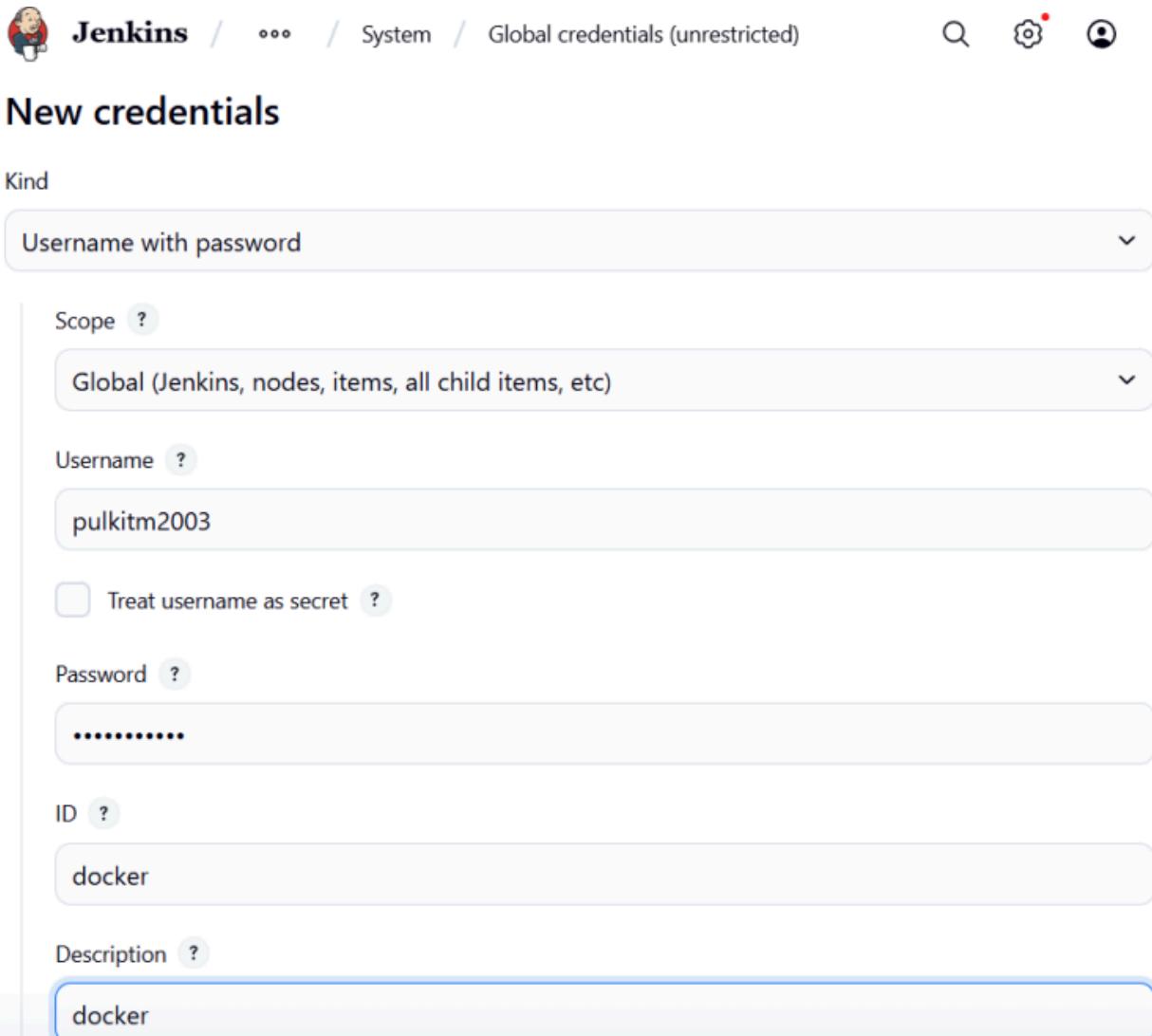
User name = your docker username

Password = your docker hub password

Id = docker

Description = docker

Click create



The screenshot shows the Jenkins Global credentials configuration page. The URL is `Jenkins / System / Global credentials (unrestricted)`. The page title is "New credentials". The "Kind" dropdown is set to "Username with password". The "Scope" dropdown is set to "Global (Jenkins, nodes, items, all child items, etc)". The "Username" field contains "pulkitm2003". The "Treat username as secret" checkbox is unchecked. The "Password" field contains a redacted password. The "ID" field contains "docker". The "Description" field contains "docker". A blue selection bar is visible under the "Description" field.

Add global credentials for ansible too

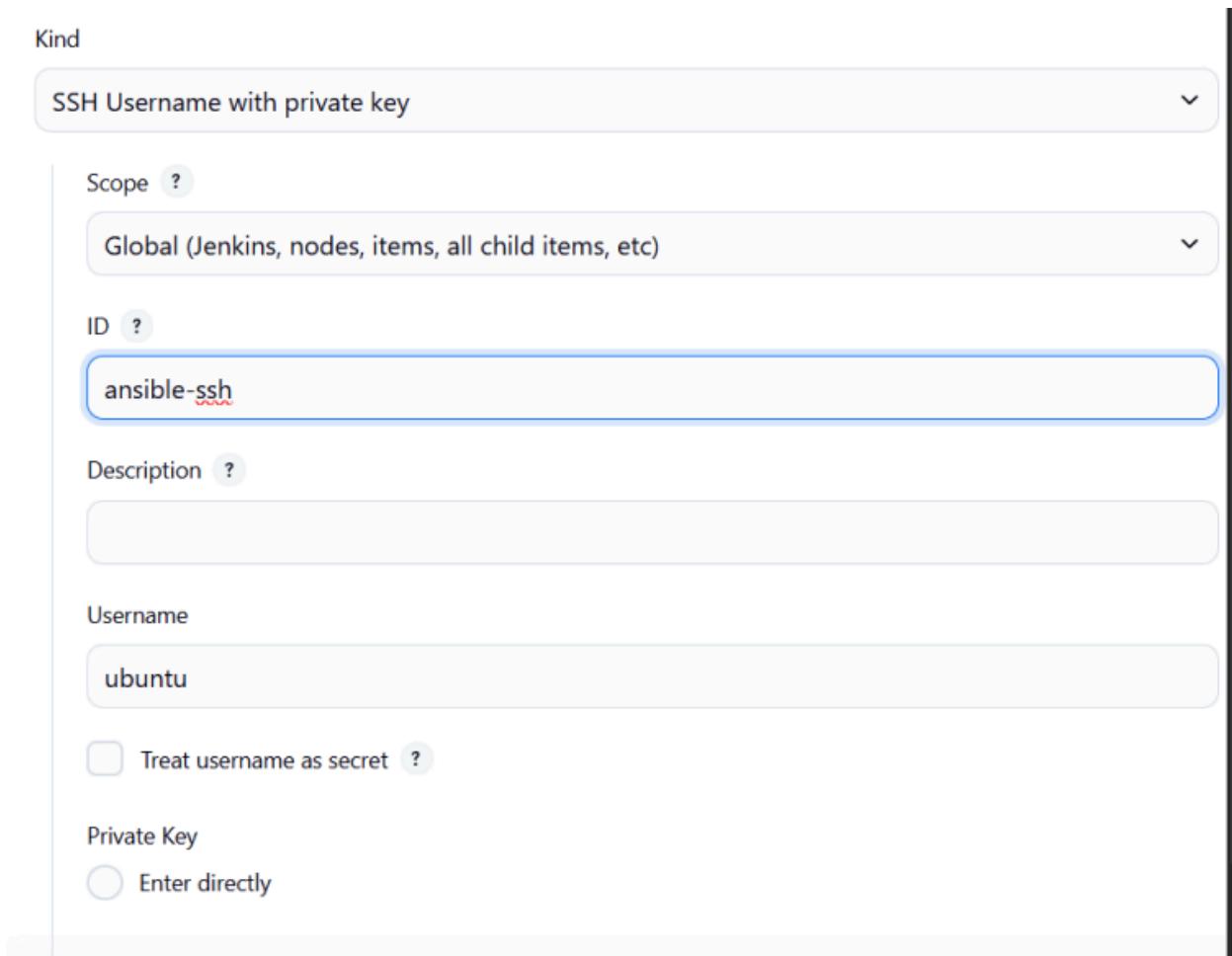
Kind : SSH Username with private key

Scope : Global

ID : ansible-ssh

Username : ubuntu

Click on Private key enter directly and enter the private key used to make master ec2



The screenshot shows the configuration for a Jenkins job. The 'Kind' dropdown is set to 'SSH Username with private key'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'ID' field contains 'ansible-ssh'. The 'Description' field is empty. The 'Username' field contains 'ubuntu'. The 'Treat username as secret' checkbox is unchecked. The 'Private Key' section has a radio button for 'Enter directly'.

Kind

SSH Username with private key

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

ID ?

ansible-ssh

Description ?

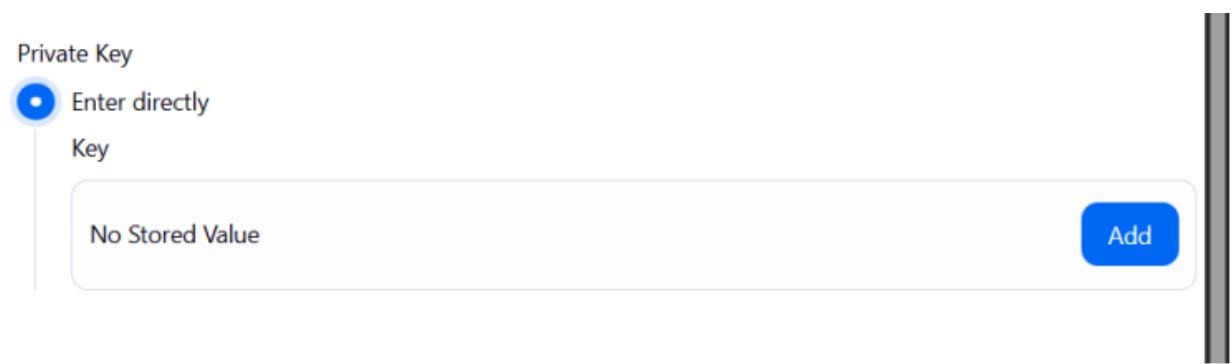
Username

ubuntu

Treat username as secret ?

Private Key

Enter directly



Add Credentials for AWS access key

Kind : Secret key

Scope : Global

Secret : <add_accesskey>

ID : aws-access-key

New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

aws-access-key

Description ?

Create

Add Credentials for AWS Secret access key

Kind : Secret key

Scope : Global

Secret : <add_Secret_accesskey>

ID : aws-secret-key



Jenkins

/ ...

/ System

/ Global credentials (unrestricted)



New credentials

Kind

Secret text

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID ?

aws-secret-key

Description ?

Create

Click on create

T	P	Store ↓	Domain	ID	Name
		System	(global)	ansible-ssh	ubuntu
		System	(global)	aws-access-key	aws-access-key
		System	(global)	aws-secret-key	aws-secret-key
		System	(global)	dockerhub-cred-id	pulkitm2003/***** (dockerhub-cred-id)

Stores scoped to Jenkins

c) go to → Manage Jenkins → Tools (under system configuration) →

click on Add jdk

name: jdk17

select install automatically

under installer dropdown

select “install from adoptium.net”

version? : select jdk-17.0.8.1+1 version

add docker

name: docker

select install automatically

add installer

Download from docker.com

add ansible
name: ansible

leave as it is

click on apply and save

Step 6. Creating Dynamic Inventory

a) Give different names

since all our nodes have same names thus we need to give them different names to differentiate here we will use the **tag** we gave earlier **Environment = dev**

vi tags.sh

```
# Fetch instance IDs that match Environment=dev  
and Role=web  
  
instance_ids=$(aws ec2 describe-instances \  
    --filters "Name>tag:Environment,Values=dev"  
    "Name=instance-state-name,Values=running" \  
    --query  
    'Reservations[*].Instances[*].InstanceId' \  
    --output text)
```

```
# Sort instance IDs deterministically  
  
sorted_ids=$(echo "$instance_ids" | tr '\t'  
'\n' | sort))
```

```
# Rename instances sequentially  
  
counter=1  
  
for id in "${sorted_ids[@]}"; do  
    name=<name_you_want>-$(printf "%02d"  
$counter)"  
    echo "Tagging $id as $name"
```

```
aws ec2 create-tags --resources "$id" \
--tags Key=Name,Value="$name"
((counter++))
```

done

press esc:wq

give permission and run file

```
root@ip-172-31-4-83:/home/ubuntu# chmod +x tags.sh
root@ip-172-31-4-83:/home/ubuntu# ./tags.sh
Tagging i-0d7c6b1e949a549f7 as PM-Node-01
Tagging i-0f62961bee9dc34c9 as PM-Node-02
```

	Name	Instance ID	Status	Instance Type
<input checked="" type="checkbox"/>	PM-Node-01	i-0d7c6b1e949a549f7	Running	t3.medium
<input type="checkbox"/>	Pulkit-Master	i-0f76c1dfbac8e0b85	Running	t2.large
<input type="checkbox"/>	PM-Node-02	i-0f62961bee9dc34c9	Running	t3.medium

b) Create Ansible key

```
ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
```

press enter 3 times

```
root@ip-172-31-4-83:/home/ubuntu# ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:cZKtc4W2m5zxI6BqOQurUbuntL25mPwGLihJeXCDUpM Ansible-Master
The key's randomart image is:
+---[RSA 4096]---+
|   .          |
| E           . |
| ...         o . |
| .o o       + o |
| . + o     S B = |
| o.o .    + B * |
| oo+.o ..   * o |
|=o.*o=+.    . . |
|..B*B*oo+oo |
+---[SHA256]---+
root@ip-172-31-4-83:/home/ubuntu#
```

c) Creating Yaml file that will hold IP address of node

```
mkdir inventory
```

```
vi inventory/aws_ec2.yaml
```

```
plugin: amazon.aws.aws_ec2
```

```
regions:
```

```
  - <your_region_name>
```

```
filters:
```

```
  tag:Environment: dev
```

```
  instance-state-name: running
```

```
compose:
```

```
  ansible_host: public_ip_address
```

```
keyed_groups:
```

```
  - key: tags.Name
```

```
    prefix: name
```

```
  - key: tags.Environment
```

```
    prefix: env
```

to test run this command

```
ansible-inventory -i inventory/aws_ec2.yaml  
-graph
```

```
root@ip-172-31-4-83:/home/ubuntu# ansible-inventory -i inventory/aws_ec2.yaml --graph  
graph TD  
@all  
|--@ungrouped:  
|--@aws_ec2:  
| |--ec2-65-1-86-174.ap-south-1.compute.amazonaws.com  
| |--ec2-15-207-21-76.ap-south-1.compute.amazonaws.com  
|--@name_PM_Node_01:  
| |--ec2-65-1-86-174.ap-south-1.compute.amazonaws.com  
|--@env_dev:  
| |--ec2-65-1-86-174.ap-south-1.compute.amazonaws.com  
| |--ec2-15-207-21-76.ap-south-1.compute.amazonaws.com  
|--@name_PM_Node_02:  
| |--ec2-15-207-21-76.ap-south-1.compute.amazonaws.com  
root@ip-172-31-4-83:/home/ubuntu# vi inventory/aws_ec2.yaml
```

Step 7. Connecting Ansible Master - Node

a) Creating Ansible Environment

We are setting up separate environment to isolate Ansible dependencies

Install venv module if not already present :

```
sudo apt install python3-venv -y
```

Create a virtual environment :

```
python3 -m venv ansible-env
```

Activate it :

```
source ansible-env/bin/activate
```

we can see now we have entered into ansible environment

```
root@ip-172-31-4-83:/home/ubuntu# python3 -m venv ansible-env
root@ip-172-31-4-83:/home/ubuntu# source ansible-env/bin/activate
(ansible-env) root@ip-172-31-4-83:/home/ubuntu#
```

Install required Python packages :

```
pip install boto3 botocore docker
```

b) Creating Key file

Open the key pair file in notepad and copy its content

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEpAIBAAKCAQEA2sZo41c83I7kSsHrZlAu+  
4hEeWC7gP7EtD2/I8SnWMWMnvAM  
wOXznV8STN3yvzSxdOZIHXpmPiG9Ncutu+XfRqQABKBbs7Yaqtzz2Zr0OPs6  
H1Kl  
z8ml35aPo6xaD11IZIEf0ROKhvS3RnYGmWIqL75QTRItnKlaGjQRRKxTRXV7  
za8e  
Cj9Wtro7  
+ssrNdGk7zWXWm9tvbf5JCc8/yLcMhUIbWnvx67j+pkuWxC6zLjtpTfF  
SUw2mLDDNkVQuC9yJq7sZO3alICG1UNGkUKPQqgVg0FzvvDSEiZ0hrxOcv0  
ZsbnO  
bi0EHnyjfroCNzSv3TvikRFkMkWExYSIcUV00wIDAQABAoIBAQC+UElgMzy'  
IV5M
```

Paste this content in file below

vi Krishna-New-kp.pem

(name should be same as key pair used to create master and nodes)

After pasting key value here press – esc:wq

Give read only permission

Chmod 400 Krishna-New-kp.pem

c) Give over ride permission

since while adding key in nodes nodes require permission as yes but we cannot open each node and give it permission as there are

```
multiple nodes thus we will over ride those  
permissions
```

```
vi ansible.cfg
```

```
[defaults]
```

```
inventory = ./inventory/aws_ec2.yaml
```

```
host_key_checking = False
```

```
[ssh_connection]
```

```
ssh_args = -o StrictHostKeyChecking=no -o  
UserKnownHostsFile=/dev/null
```

d) Script to connect Nodes and Master

```
vi copy-public-key.sh
```

```
#!/bin/bash
```

```
# Define vars
```

```
PEM_FILE="Krishna-New-kp.pem"
```

```
PUB_KEY=$(cat ~/.ssh/id_rsa.pub)
```

```
USER="ec2-user" # Since nodegroup is Linux

INVENTORY_FILE="inventory/aws_ec2.yaml"

# Extract hostnames/IPs from dynamic inventory

HOSTS=$(ansible-inventory -i $INVENTORY_FILE
--list | jq -r '.[_meta.hostvars] | keys[]')

for HOST in $HOSTS; do
    echo "Injecting key into $HOST"
    ssh -o StrictHostKeyChecking=no -i $PEM_FILE
$USER@$HOST "
        mkdir -p ~/.ssh && \
        echo \"\$PUB_KEY\" >> ~/.ssh/authorized_keys
        && \
        chmod 700 ~/.ssh && \
        chmod 600 ~/.ssh/authorized_keys
    "
done

give permission and run file
```

```
chmod +x copy-public-key.sh
```

```
./copy-public-key.sh
```

Step 8. Final Deployment Stage

a) Give Jenkins Permission to ansible environment

```
sudo chown -R jenkins:jenkins  
/home/ubuntu/ansible-env
```

```
sudo chmod -R 755 /home/ubuntu/ansible-env
```

```
sudo chmod 755 /home/ubuntu
```

verify

```
sudo -u jenkins ls -l  
/home/ubuntu/ansible-env/bin/activate
```

```
sudo -u jenkins bash -c  
"source /home/ubuntu/ansible-env/bin/activate &&  
echo Activated"
```

```
root@ip-172-31-4-83:/home/ubuntu/k8s# sudo -u jenkins bash -c "source /home/ubuntu/ansible-env/bin/activate && echo Activated"  
Activated
```

```
sudo usermod -aG docker jenkins
```

```
sudo systemctl restart Jenkins
```



```
# Inside ansible-env
```

```
source /home/ubuntu/ansible-env/bin/activate
```

```
pip install --upgrade pip
```

```
pip install ansible boto3 botocore kubernetes
```

```
ansible-galaxy collection install  
kubernetes.core
```

```
deactivate
```



```
[REDACTED] k8s folder and makes  
files in it
```

```
Make files on forked repository
```

```
mkdir k8s
```

```
vi k8s/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

    name: apache-deployment

spec:

    replicas: 2

    selector:

        matchLabels:

            app: apache

    template:

        metadata:

            labels:

                app: apache

        spec:

            containers:

                - name: apache
```

```
        image:  
<your_docker_id>/apachewebsite:latest # Your  
Docker Hub image  
  
        ports:  
          - containerPort: 80
```

```
vi k8s/service.yaml
```

```
apiVersion: v1  
  
kind: Service  
  
metadata:  
  name: apache-service  
  
spec:  
  type: LoadBalancer  
  
  selector:  
    app: apache  
  
  ports:  
    - protocol: TCP
```

```
    port: 80
```

```
    targetPort: 80
```

```
[REDACTED]
```

```
- name: Deploy Apache website to EKS
```

```
  hosts: localhost
```

```
  connection: local
```

```
  tasks:
```

```
    - name: Ensure AWS CLI is configured for EKS
```

```
      command: aws eks update-kubeconfig --name
<your_cluster_name> --region <your_region_name>
```

```
    - name: Deploy Apache Deployment
```

```
      kubernetes.core.k8s:
```

```
        state: present
```

```
        namespace: default
```

```
src: k8s/deployment.yaml
```

```
- name: Deploy Apache Service
```

```
kubernetes.core.k8s:
```

```
    state: present
```

```
    namespace: default
```

```
src: k8s/service.yaml
```

```
- name: Verify deployment pods
```

```
command: kubectl get pods -o wide
```

```
register: pod_status
```

```
- debug:
```

```
var: pod_status.stdout_lines
```

```
pipeline {
```

```
    agent any
```

```
    environment {  
  
        DOCKER_IMAGE =  
        "<docker_id>/apachewebsite:latest"  
    }
```

```
    stages {  
  
        stage('Checkout SCM') {  
  
            steps {  
  
                checkout scm  
  
            }  
        }  
    }
```

```
        stage('Docker Build & Push') {  
  
            steps {  
  
                script {
```

```
withDockerRegistry([credentialsId:
```

```
'dockerhub-cred-id', url:  
'https://index.docker.io/v1/']) {  
  
    sh '''  
  
    echo "Building  
Docker image..."  
  
    docker build -t  
$DOCKER_IMAGE .  
  
  
  
    echo "Pushing Docker  
image to DockerHub..."  
  
    docker push  
$DOCKER_IMAGE  
  
    '''  
  
}  
  
}  
  
}  
  
}  
  
stage('Prepare Kubernetes Manifests') {
```

```
        steps {
            sh '''
                echo "Copying Kubernetes
manifests into Jenkins workspace..."
                mkdir -p k8s
                cp
                /home/ubuntu/k8s/service.yaml k8s/service.yaml
                cp
                /home/ubuntu/k8s/deployment.yaml
                k8s/deployment.yaml
            '''
        }
    }
```

```
stage('Deploy to EKS via Ansible') {
    steps {
        withCredentials([
            string(credentialsId:
            'aws-access-key', variable:
            'AWS_ACCESS_KEY_ID'),
            string(credentialsId:
            'aws-secret-key', variable:
            'AWS_SECRET_ACCESS_KEY'),
            string(credentialsId:
            'aws-region', variable:
            'AWS_DEFAULT_REGION')
        ])
    }
}
```

```
        string(credentialsId:  
'aws-secret-key', variable:  
'AWS_SECRET_ACCESS_KEY')  
  
    ]) {  
  
        sh '''  
  
        echo "Exporting AWS  
credentials..."  
  
        export  
AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID  
  
        export  
AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY  
  
        export  
AWS_DEFAULT_REGION=ap-south-1  
  
        echo "Activating Ansible  
virtual environment..."  
  
        .  
/home/ubuntu/ansible-env/bin/activate  
  
        echo "Running Ansible  
playbook for Kubernetes deployment..."
```

```
ansible-playbook  
deploy-k8s.yaml
```

```
    ...  
}  
}  
}
```

```
stage('Verify Deployment') {  
    steps {  
        echo "Stage skipped if  
deployment failed"  
    }  
}
```

```
    post {  
        success {  
            echo "✅ Deployment succeeded!"  
        }
```

```
        }

    failure {
        echo "✖ Deployment failed!"

    }

}
```



Jenkins

+ New Item

Build History

Project Relationship

Check File Fingerprint

Build Queue



No builds in the queue.

Build Executor Status

0/2



Enter name -> Select Pipeline option -> click on OK



New Item

Enter an item name

Apache-proj

Select an item type



Freestyle project

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Scroll down to Pipeline section

Select Definition -> Pipeline script from SCM

Select SCM as Git

Under Repositories Add <your repository url>

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/pulkit2003/apachewebsite

Credentials ?

- none -

+ Add

Advanced ▾

The screenshot shows the Jenkins Pipeline configuration page. Under the 'Pipeline script from SCM' section, 'Git' is selected. In the 'Repositories' section, there is one repository defined with the URL 'https://github.com/pulkit2003/apachewebsite'. The 'Credentials' dropdown is set to '- none -'. There is also a '+ Add' button for adding new credentials. At the bottom, there is an 'Advanced' dropdown menu.

Select Master as Branch

Script path as Jenkinfile

Then click Apply and Save

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? X

*/master

Add Branch

Repository browser ?

(Auto) ▼

Additional Behaviours

Add ▾

Script Path ?

Jenkinsfile

b) Run pipeline

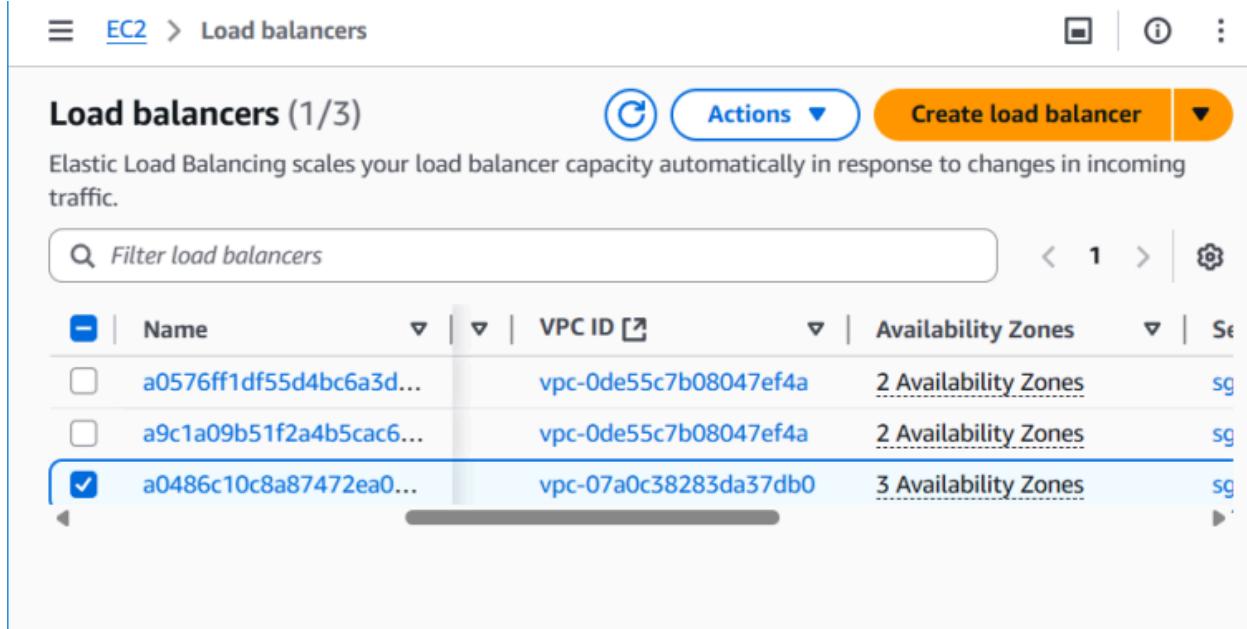
Stage View



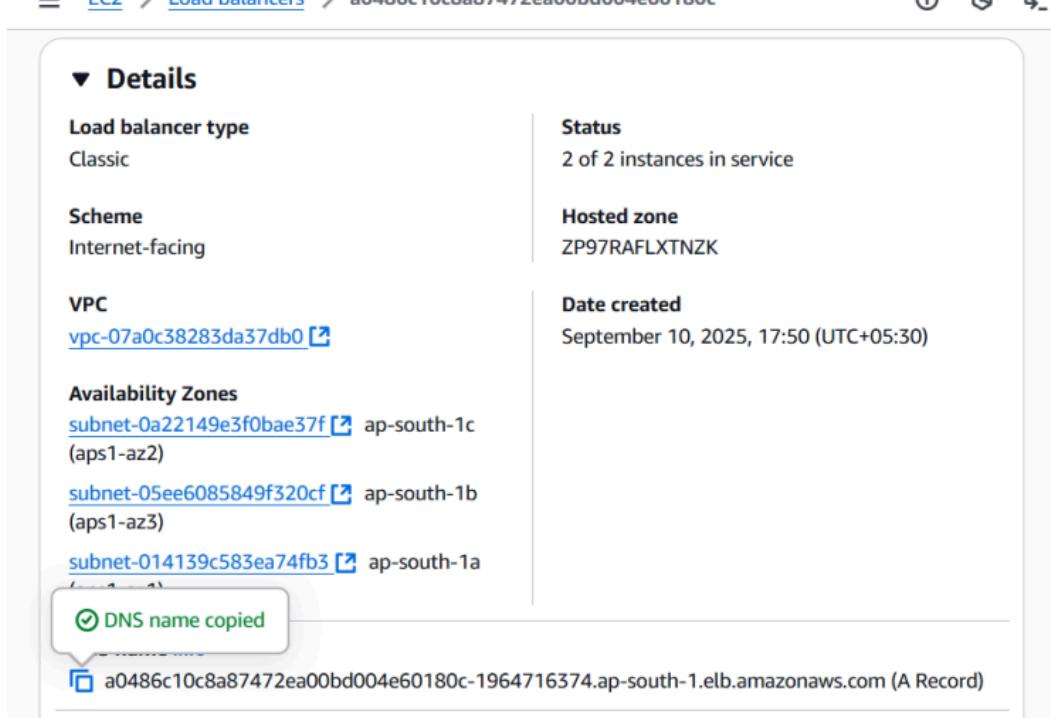
Permalinks

Step 10. Confirmation

a) Open Load Balancer and select your Load Balancer that has been created Copy the DNS link and open on Browser



Name	VPC ID	Availability Zones	SG
a0576ff1df55d4bc6a3...	vpc-0de55c7b08047ef4a	2 Availability Zones	sg
a9c1a09b51f2a4b5cac6...	vpc-0de55c7b08047ef4a	2 Availability Zones	sg
a0486c10c8a87472ea0...	vpc-07a0c38283da37db0	3 Availability Zones	sg



EC2 > Load balancers > a0486c10c8a87472ea00bd004e60180c

Details

Load balancer type	Status
Classic	2 of 2 instances in service
Scheme	Hosted zone
Internet-facing	ZP97RAFLXTNZK
VPC	Date created
vpc-07a0c38283da37db0	September 10, 2025, 17:50 (UTC+05:30)
Availability Zones	
subnet-0a22149e3f0bae37f ap-south-1c (aps1-az2)	
subnet-05ee6085849f320cf ap-south-1b (aps1-az3)	
subnet-014139c583ea74fb3 ap-south-1a	

DNS name copied

[a0486c10c8a87472ea00bd004e60180c-1964716374.ap-south-1.elb.amazonaws.com \(A Record\)](#)

The site is running

